ULTRIX

Guide to System Crash Recovery

Order Number: AA-ME94B-TE
June 1990

Product Version: ULTRIX Version 4.0 or higher

This manual describes the crash dump process and describes how to obtain crash dump files.

digital equipment corporation
maynard, massachusetts

Contents

About This Manual

This guide provides information on recovering from a system crash using the ULTRIX utilities. It also presents guidelines from which you can develop specific crash recovery procedures for your site.

Audience

The *Guide to System Crash Recovery* is written for the person responsible for managing and maintaining an ULTRIX system. It assumes that this individual is familiar with ULTRIX commands, the system configuration, the system's controller/drive unit number assignments and naming conventions, and an editor such as `vi` or `ed`. You do not need to be a programmer to use this guide.

Organization

This manual consists of two chapters:

Chapter 1	System Crash Recovery Explains what the system does when a crash occurs.
Chapter 2	Forcing a Crash Dump Describes how to obtain the crash dump files when the crash dump routine does not execute properly.

Related Documents

You should have the following documentation:

- The hardware documentation for your system and peripherals
- The *Guide to Configuration File Maintenance* for information on swap space
- The *Guide to System Environment Setup* for information on maintaining administrative files

Conventions

The following conventions are used in this manual:

%	The default user prompt is your system name followed by a right angle bracket. In this manual, a percent sign (%) is used to represent this prompt.

#	A number sign is the default superuser prompt.
`user input`	This bold typeface is used in interactive examples to indicate typed user input.
`system output`	This typeface is used in interactive examples to indicate system output and also in code examples and other screen displays. In text, this typeface is used to indicate the exact name of a command, option, partition, pathname, directory, or file.
UPPERCASE lowercase	The ULTRIX system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.
rlogin	In syntax descriptions and function definitions, this typeface is used to indicate terms that you must type exactly as shown.
macro	In text, bold type is used to introduce new terms.
filename	In examples, syntax descriptions, and function definitions, italics are used to indicate variable values; and in text, to give references to other documents.
[ ]	In syntax descriptions and function definitions, brackets indicate items that are optional.
{ | }	In syntax descriptions and function definitions, braces enclose lists from which one item must be chosen. Vertical bars are used to separate items.
. . .	In syntax descriptions and function definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
. . .	A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.
`cat(1)`	Cross-references to the *ULTRIX Reference Pages* include the appropriate section number in parentheses. For example, a reference to `cat(1)` indicates that you can find the material on the `cat` command in Section 1 of the reference pages.
RETURN	This symbol is used in examples to indicate that you must press the named key on the keyboard.
CTRL/*x*	This symbol is used in examples to indicate that you must hold down the CTRL key while pressing the key *x* that follows the slash. When you use this key combination, the system sometimes echoes the resulting character, using a circumflex (^) to represent the CTRL key (for example, ^C for CTRL/C). Sometimes the sequence is not echoed.

ESC X	This symbol is used in examples to indicate that you must press the first named key and then press the second named key. In text, this combination is indicated as ESC-X.

System Crash Recovery 1

This chapter explains what happens during a system crash, how the dump process works, and how to maintain file system consistency when the system reboots. In addition, this chapter describes how to save the dump files and provides you with the commands you use to analyze them.

1.1 System Crashes and the Dump Process

The system monitors its own internal status and performs a number of internal consistency checks. If an internal check shows inconsistencies, the system prints panic messages to the console and then crashes. The panic messages help you determine the cause of the crash.

Prior to a system crash, but after a panic message is displayed, the system updates all file system information. The system then performs a core dump of selected physical memory pages to the dump device specified in the system configuration file. The following pages are dumped during a crash:

- All kernel image pages (text/data/bss/valloc)
- All kernel memory allocator pages (kmalloc data)
- All active process context pages (active user areas)
- All inactive process context pages (inactive user areas)
- All the active and inactive process page table pages

By selectively choosing the pages that are to be dumped, only a minimum amount of disk space is needed to handle a system crash. Section 1.1.1 and Section 1.1.2 describe how to calculate the dump partition sizes for VAX or RISC processors.

If, for some reason, a full dump is necessary, you can specify the `FULLDUMPS` options in the system configuration file. This option enables full crash dumps. Note that you must also increase the size of the dump partition to the size of physical memory before reconfiguring your system.

After the system dumps the raw memory image, the system reboots itself and invokes the `/etc/fsck` command to check for file system inconsistencies.

1.1.1 Calculating the Dump Partition on a VAX Processor

On VAX machines, assuming the maximum physical memory size is 512 megabytes and the maximum number of users is 256, you should allocate a dump partition of 34 megabytes. This partition size is based on the following estimates:

- Kernel image pages, 4 megabytes
- Kernel memory allocator pages, 10 megabytes

- Active and inactive process context pages, 16 megabytes
- Active and inactive process page table pages, 4 megabytes

Use the following table to estimate partition sizes for VAX machines based on physical memory size and maximum number of users:

PHYSMEM	MAXUSERS	Dump Partition
16 megabytes	16	10 megabytes
32 megabytes	32	12 megabytes
64 megabytes	64	16 megabytes
128 megabytes	128	26 megabytes
256 megabytes	128	26 megabytes
512 megabytes	256	34 megabytes

1.1.2 Calculating the Dump Partition on RISC Processors

On RISC machines, assuming the maximum physical memory size is 512 megabytes and the maximum number of users 256, you should allocate a dump partition of 48 megabytes. This partition size is based on the following estimates:

- Kernel image pages, 4 megabytes
- Kernel memory allocator pages, 16 megabytes
- Active and inactive process context pages, 16 megabytes
- Active and inactive process page table pages, 12 megabytes

Use the following table to estimate partition sizes for RISC machines based on physical memory size and the maximum number of users:

PHYSMEM	MAXUSERS	Dump Partition
16 megabytes	16	14 megabytes
32 megabytes	32	20 megabytes
64 megabytes	64	28 megabytes
128 megabytes	128	40 megabytes
256 megabytes	128	40 megabytes
512 megabytes	256	48 megabytes

1.2 Maintaining File System Consistency After a Crash

This section discusses how file system inconsistencies occur, how they are corrected during daily operations, and how to proceed if the `fsck` command cannot correct inconsistencies during the reboot process.

1.2.1 Identifying File System Inconsistencies

Before the system crashes, it tries to update all file system information. The system keeps copies in memory of the information for all active file systems. The system's in-memory buffer cache contains copies of the recently used free block lists, free inode lists, modified data blocks, and the modified inodes of the mounted file systems. It also keeps all the modified superblocks of the mounted file systems.

To coordinate the changes recorded in these in-memory copies with the permanent summary information, the system periodically updates all file system information. That is, the `update` command executes every 30 seconds and invokes the `sync` system routine. However, when the system crashes, the disk-resident file system information may not be completely updated. If this occurs, inconsistencies exist between the summary information and the actual status of the file system. These can be corrected during the reboot process.

1.2.2 Invoking the fsck Command Using /etc/rc

Unless your system has a clean shutdown, the `fsck` command checks the file systems for inconsistencies each time the system reboots. The `/etc/rc` file automatically invokes the `fsck` command to check and correct those inconsistencies that can be easily fixed.

If the `fsck` command encounters inconsistencies that cannot be easily corrected, `/etc/rc` exits multiuser startup and your system remains in single-user mode. You are instructed to run the `fsck` command manually. This allows you to immediately correct specific file system inconsistencies.

1.2.3 Interactively Executing the fsck Command

The `fsck` command checks your file systems when invoked for interactive execution. As it encounters each inconsistency, the `fsck` command displays a diagnostic message that indicates the type of inconsistency found and prompts you for a response to the displayed corrective action. You must answer either `yes` or `no` to this prompt.

If you answer `yes` to a corrective action prompt, the `fsck` command attempts to implement the corrective action. In addition, if necessary, the `fsck` command relinks all allocated but unlinked files to the `lost+found` directory for the appropriate file system. To relink a file, the `fsck` command uses the file's inode number as its name.

If the `fsck` command relinks a file, you should determine the file's owner and the directory in which it belongs, as follows:

1. Use the `ls` command with the `-i` option to gather information about the file's inode number.
2. Use the `file` command to determine the file type.
3. Contact the owner of the file and determine which directory the file belongs in. You can then move the file from the `lost+found` directory to the correct directory.

Note

> The fsck command requires a lost+found directory in each file system. The newfs command creates this directory in each file system. However, if one of these directories is inadvertently removed during operations, use the mklost+found command to create this directory.

If you answer no to the corrective action prompt, the fsck command continues to check for other inconsistencies and creates a summary that enables you to determine your own corrective measures. If the fsck command can provide alternate corrective actions, it continues to prompt you for a response.

If the fsck command tells you to reboot the system after correcting the root file system, halt or reset. This returns you to the console prompt and allows you to boot again. For information on how to halt or reset your processor, see the hardware documentation for your processor.

As the system reboots to multiuser mode, the fsck command continues to check and correct inconsistencies in other file systems.

For more information, see the fsck(8) and mklost+found(8) commands in the *ULTRIX Reference Pages*.

Note

> The fsck command has made the other file system maintenance commands obsolete by combining their functions. However, for further information, see clri(8), dcheck(8), dumpfs(8), icheck(8), and ncheck(8) in the *ULTRIX Reference Pages*.

1.2.4 Restoring Pseudoterminals Invoked by /etc/rc.local

After a system crash, ownership and permissions of pseudoterminals are restored to normal by the /etc/rc.local file. When the system returns to multiuser mode, ownership is root and permissions are 666 (read/write access).

1.3 Generating Crash Dump Files

To determine why a crash occurred, you must generate crash dump files that you can analyze. To create the crash dump files, use the savecore command. The savecore command saves the kernel image in the file vmcore, the namelist in vmunix, and the errorlog entries.

The following sections discuss how to invoke the savecore command during the reboot process or how to manually invoke the savecore command.

1.3.1 Generating Crash Dump Files During the Reboot Process

To generate crash dump files during the reboot process, include a savecore entry in the /etc/rc.local file using the following format:

/etc/savecore *options dirname*

The options are as follows:

–c Clears the core dump. If a core dump has been corrupted in a way that does not allow the savecore command to safely save the dump files, this

command removes the core dump from the system. Use caution when specifying this option, because the core dump cannot be retrieved after it has been removed.

–d *dumpdev dumplo*
: Specifies the dump device and dump device offset when running savecore on a system image other than the currently running system image. The savecore command assumes that the running system is /vmunix and it reads the dump device and dump device offset from /dev/kmem. If the dump device and the dump device offset differ in the system image that crashed, this option can help determine the correct dump device and dump device offset.

-e
: Moves only the error logger buffer into a file. If this option is specified, the kernel image and the namelist image are not saved.

-f
: Takes the *i corefile* name as the file from which to extract the crash dump data instead of the default dump device. This option is only used for diskless workstations.

The *dirname* can be any directory (file system) that has enough space to contain the dump files. The default directory is /usr/adm/crash. If you specify a directory other than the default, create that directory before specifying it in your savecore entry.

Note

If the directory specified by the savecore entry does not contain enough space to store vmcore and vmunix, the savecore command dumps as much as possible and then issues the following message:

```
write:  No space left on device
```

Unless the memory dump is overwritten because of system swap activity, you can obtain a full dump by creating space in the dump file directory, and then manually running the savecore command.

1.3.2 Generating Crash Dump Files Manually

To manually begin a crash dump, boot the system to single-user mode, then invoke the savecore command as follows:

/etc/savecore *dirname*

You must replace *dirname* with the name of a directory (file system) large enough to contain the dump files. The default directory is /usr/adm/crash.

1.4 Creating a Copy of the Dump Files

To create a copy of the dump files, you must use the dd command. This command has an option that enables you to create sparse output files. Remember that the vmcore file created by savecore is a sparse file. If you copy this file using a command such as cp, it will expand and possibly use up system file space. Hence, use the dd command to copy the sparse files and you can reserve file system space. See the *ULTRIX Reference Pages* for more information.

After you copy the dump files, you should remove them from the directory (file system) to conserve space. For example, use the rm command to remove the files as

follows:

```
# rm /usr/adm/crash/vmunix.1 /usr/adm/crash/vmcore.1
```

For further information, see the dd(1), rm(1), and tar(1) commands in the *ULTRIX Reference Pages.*

1.5 Examining the Dump Files

The crash dump files help determine the cause of a system crash. To examine the crash dump or partial dump file, use the adb or dbx commands, or the crash utility as follows:

- On VAX processors, use the adb command to examine the dump files.
- On RISC processors, use the dbx command to examine dump files.
- The crash utility can be used on either VAX or RISC processors.

When analyzing a partial crash dump, the vmcore.*n* file created by the savecore command is a sparse file. Hence, the vmcore.*n* file contains spaces for all the pages that were not dumped during the crash. If you try to examine a page in the vmcore.*n* file that was not dumped, it returns all zeros.

Forcing a Crash Dump 2

This chapter describes the procedures you must follow to force a memory dump on a VAX or RISC processor.

Usually, the system reboots itself after a crash occurs. If the system does not reboot, a condition may exist that prevents the crash dump routine from executing properly. For example, the system cannot execute the crash dump routine when an invalid interrupt stack in the kernel address space exists. Should this condition exist, you must do the following:

- For VAX processors, you can try to manually start a memory dump, force a segmentation fault, or initialize the processor.

  Each successive method yields less information about the cause of a crash, because more of the machine state is altered. As you move through each method, you can assume that the cause of the crash is more serious. Starting a crash dump routine manually is the preferred course of action. If you cannot manually start a crash dump, force a segmentation fault. Avoid initializing the processor, unless an attempt to force a segmentation fault does not work. See Sections 2.1 through 2.3 for instructions.

- For RISC processors, you can manually start a memory dump. If this method is not successful, the memory dump was corrupted and cannot be recovered. See Section 2.4 for instructions.

2.1 Starting the Crash Dump Routine Manually on VAX Processors

When you start a crash dump manually, the current machine state is not affected. This is the suggested course of action. The following steps let you manually start a memory dump on a VAX processor:

1. Enter console mode by halting your processor. The hardware documentation for your processor tells you how to enter console mode.

2. Examine the program counter (PC) that contains the address of the next instruction to be executed and stored in general register F. For example:

   ```
   >>>E/G F
     G 0000000F  80001EAD
   ```

3. Examine the process status longword (PSL) that contains the execution state of the processor at the time that the crash occurred. For example:

   ```
   >>> E PSL
     M 00000000  04C10004
   ```

   See the *VAX Hardware Handbook* for more information on the bit meanings in the PSL.

4. Set the PSL to interrupt stack with an interrupt priority level (IPL) 31. This sets the processor to run on the interrupt stack and blocks interrupts. For example:

   ```
   >>>D PSL 041F0000
   ```

5. Find the address of the dump routine by examining the fourth physical longword of the restart parameter block (RPB). For example:

   ```
   >>>E/P/L 4
     P00000004 00001E00
   ```

   The system displays the physical address location of the dump routine.

6. Start execution of the dump routine. For example:

   ```
   >>>S 8nnnnnnn
   ```

   Note that bit 31 has been changed to reflect the virtual address of the crash dump routine obtained in step 6. This is a necessary change because the processor is still set to run in virtual memory mode.

The system should execute the dump routine, reboot itself, and place the crash dump files in the directory (file system) specified in the `/etc/rc.local` file.

To analyze the crash dump, use the `adb` and the `nm` commands. See `adb`(1) and `nm`(1) in the *ULTRIX Reference Pages* for more information.

2.2 Forcing a Segmentation Fault on VAX Processors

If you cannot manually start the crash dump routine, set up a condition that forces a segmentation fault and instructs the processor to continue. To force a segmentation fault, you must set the program counter (PC) to an address that is outside of the process address space, such as PC –1. This causes the processor to synchronize the disks; however, some of the current machine state is changed.

Before you set the PC to an invalid address such as –1, examine the PC and stack pointers, because these change when you force the segmentation fault.

Use the following steps to force a segmentation fault:

1. Enter console mode by halting your processor. The hardware documentation for your processor describes how to enter console mode.

2. Examine the PC stored in general register F. For example:

   ```
   >>>E/G F
     G 0000000F  80001EAD
   ```

3. Examine the process status longword (PSL). For example:

   ```
   >>>E PSL
     M 00000000  04C10004
   ```

4. Display and record the kernel stack pointer (KSP), because this changes when you force a segmentation fault. The KSP is stored in internal register 0. For example:

   ```
   >>>E/I 0
     I 00000000  7FFFFDAC
   ```

5. Display and record the user stack pointer (USP), because this changes when you force a segmentation fault. The USP is stored in internal register 3. For example:

```
>>>E/I 3
  I 0000003  7FFFE2F4
```

6. Display and record the interrupt stack pointer (ISP), because this changes when you force a segmentation fault. The ISP is stored in internal register 4. For example:

```
>>>E/I 4
  I 00000004  80000C00
```

7. Set the PC to –1. For example:

```
>>>D/G F FFFFFFFF
```

8. Set the PSL to interrupt priority level (IPL) 31 to block interrupts. For example:

```
>>>D PSL 001F0000
```

9. Instruct the processor to continue. For example:

```
>>>C
```

The system should execute the crash dump routine, reboot itself, and place the crash dump files in the directory (file system) specified in the `/etc/rc.local` file.

To analyze the crash dump, use the `adb` and the `nm` commands. See `adb`(1) and `nm`(1) in the *ULTRIX Reference Pages* for more information.

2.3 Initializing a VAX Processor

If neither of the previous methods force a crash dump, you may be able to do so by initializing the processor before starting the dump routine. This action sets the processor to a known state by setting the PSL to run on the interrupt stack and the IPL to 31. In addition, the processor disables memory mapping.

Using this method, however, affects more of the machine state. Depending on your processor, the initialization may corrupt the following:

- The interrupt stack pointer (ISP)
- The kernel stack pointer (KSP)
- The P0 space base register (P0BR)
- The P0 space length register (P0LR)
- The P1 space base register (P1BR)
- The P1 space length register (P1LR)

See the *VAX Architecture Handbook* for more information on these address spaces.

Use the following steps to initialize the processor:

1. Enter console mode by halting or resetting your processor. The hardware documentation for your processor describes how to enter console mode.
2. Examine the restart parameter block (RPB) to obtain the dump address. For example:

```
>>>E/P/L 4
  P 00000004   00001E00
```

The processor displays the dump address.

3. Initialize the processor. For example:

   ```
   >>>I
   ```

4. Start execution of the dump. For example:

   ```
   >>>S 1E00
   ```

   When you initialize the processor, you must specify the physical address of the dump routine, because the processor is not running in virtual memory mode.

This method should cause the system to produce a crash dump, reboot itself, and place the crash dump data in the directory (file system) specified in the /etc/rc.local file. If this method does not yield the crash dump data, the memory dump was corrupted and cannot be retrieved.

2.4 Starting the Crash Dump Routine Manually on RISC Processors

When you start a crash dump manually, the current machine state is not affected. The following steps let you manually start a memory dump on a RISC processor:

1. Enter console mode by resetting your processor. The hardware documentation for your processor tells you how to enter console mode.

   Note

   On a DS3100, when you enter console mode by resetting the processor, memory is automatically reinitialized. To preserve memory, you must set the bootmode to debug. For example:

   ```
   >>>setenv bootmode d
   ```

   Note that if the system fails to do a memory dump at some later time, you do not have to reset the bootmode to debug.

2. Find the address of the kernel dump routine by examining the second long word of the ULTRIX save state area. For example:

   ```
   >>>e -w 0x8001f804
   ```

   See /sys/machine/mips/entrypt.h, which contains the format of the save state area.

3. Start execution of the dump routine with the address obtained from the examine command:

   ```
   >>>go 0x8nnnnnnn
   ```

   If the system was in multiuser mode when you reset the processor, the dump occurs silently and messages are not printed. The memory dump takes several minutes to complete and then the console prompt reappears.

   You can also start the dump by typing the fixed address of the coredump routine which calls the kernel dump routine. The command is as follows:

   ```
   >>>go 0X80030008
   ```

4. Reinitialize the system and then reboot the processor.

```
>>>init
>>>boot
```

Note that when the system has been shut down, halted, or reset to console mode and the `bootmode` is set to debug, the `init` (initialize) command must be typed before you type the `boot` or `auto` command. If you do not initialize the system, the system boot may fail.

The crash dump data is placed in the directory (file system) specified in the `/etc/rc.local` file.

Index

R

S

V

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	———	Local Digital subsidiary or approved distributor
Internal*	———	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX
Guide to
System Crash Recovery
AA-ME94B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? ______________________________

What do you like best about this manual? ______________________________

What do you like least about this manual? ______________________________

Please list errors you have found in this manual:

Page Description

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using? __________

Name/Title ______________________ Dept. __________

Company ______________________ Date __________

Mailing Address ______________________

__________ Email __________ Phone __________

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

Reader's Comments

ULTRIX
Guide to
System Crash Recovery
AA-ME94B-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? ______________________________

What do you like best about this manual? ______________________________

What do you like least about this manual? ______________________________

Please list errors you have found in this manual:

Page	Description

Additional comments or suggestions to improve this manual:

What version of the software described by this manual are you using? ____________

Name/Title ______________________ Dept. ____________

Company ______________________ Date ____________

Mailing Address ______________________

____________ Email ____________ Phone ____________

Do Not Tear – Fold Here and Tape

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

Do Not Tear – Fold Here

Cut
Along
Dotted
Line

digital